A Flow Control Framework for Improving Throughput and Energy Efficiency in CSMA/CA based Wireless Multihop Networks

Jaya Shankar Pathmasuntharam[1,2], Amitabha Das[2], Prasant Mohapatra[3]
[1]*Networking Department, Communications and Device Division, Institute for Infocomm Research, Singapore*
[2]*School of Computer Engineering, Nanyang Technological University, Singapore*
[3]*Department of Computer Science, University of California, Davis. Davis, CA 95616, USA*

***Abstract*— In multihop wireless networks where a random access MAC scheme such as CSMA/CA is used, nodes greedily compete in a distributed manner and are unaware of the interference they cause to other surrounding nodes. In these networks, excessive interference at a receiver or a potential forwarding node causes severe blocking and reduction in throughput. In addition, the unbalanced interference experienced at a particular node can force the node to consume more time receiving packets rather than sending them, resulting in dropped packets due to buffer overflow. We discuss a novel flow control framework for regulating the transmission and improving the overall throughput of multihop wireless networks based on CSMA/CA MAC protocol. The framework prevents congestion, reduces packet loss and is attractive because per-flow information at each node is kept to a minimum. The techniques used to improve throughput include a hop-by-hop, hybrid rate and window based flow control scheme that paces the transmission of frames such that competition between frames originating from the same flow is reduced. In a general chain topology, the framework shows that throughput and energy efficiency can be increased by factors of 2.47 and 2.4, respectively, when a single flow is transmitted as fast as possible along the chain. In the high fan-in chain setup, where multiple single-hop flows are forwarded into a chain of nodes, the energy efficiency and throughput improvement factors can be as high as 5.14 and 5.58, respectively.**

Keywords-component; Flow control, hop-by-hop, rate control, static window, multihop wireless networks, CSMA/CA

I. Introduction

In multihop wireless networks, the media access control (MAC) layer plays an important role in regulating the amount of traffic that can traverse via a node. The use of a pure TDMA approach in multihop networks requires many practical problems such as synchronization and scheduling overhead to be solved first. Not surprisingly, a random access method such as CSMA/CA is still popular in ad hoc and sensor networks due to its simplicity [10] [23]. Recent protocols such as IEEE 802.15.3 [9] and IEEE 802.15.4 [9] are focusing on a hybrid TDMA and CSMA approach. Despite the popularity of CSMA/CA based MAC protocols, its performance in multihop wireless network is known to be poor [11]. In this paper, we carefully study some of the main contributing factors of throughput degradation in CSMA/CA based MAC protocol. In addition, we also analyze the congestion problem, and show its relationship to the inefficiency of the MAC scheme. Thus the primary focus of this paper is to develop a framework for flow control for CSMA/CA based multihop wireless networks, which can improve the performance while conserving energy.

Flow and congestion control has been well researched in the context of wired networks [1] [2] [3] [4]. Over the past few years, it has been receiving wide attention in the context of wireless ad hoc networks [5] [6] [24-28], and in sensor networks [7] [8]. Unlike the wired access medium, the wireless access medium, which is typified by hidden terminal problems, broadcast transmissions, and random access methods, has significant contributions to the congestion problem. In a typical wireless multihop network, each node has a single transceiver, which is used to compete with other nodes in the shared medium to access the channel. Besides sending its own packets, each node also serves to forward other transiting packets. If a transceiver is captured more often by other nodes for receiving incoming transit packets rather than being allowed to send packets out, then congestion at that node will occur. In the wired medium, such a phenomenon does not exist since each link operates independently of others. Congestion, which leads to dropped packets at the downstream nodes, is obviously detrimental to the energy efficiency of an ad hoc or sensor network because of the wasted energy used to forward packets along multiple hops.

In this paper, we examine and design several techniques that span different layers of the traditional protocol stack to address the performance of the CSMA/CA MAC based multihop network. These include a novel hop-by-hop, hybrid rate and window based flow control scheme and an enhanced CSMA/CA MAC protocol. Collectively, we term the entire work as a flow control framework. The choice of a hop-by-hop flow control scheme over an end-to-end scheme is straightforward since feedback is faster resulting in shorter response time to congestion. In addition, it is easier to design a loss less scheme that can complement the modest energy level of the nodes.

As a precursor to the design of the proposed framework, we identify the factors that affect (some of which are new) the throughput of CSMA/CA-based multihop wireless networks. The impacts of these factors are quantified experimentally. An enhanced CSMA/CA-based scheme is developed to mitigate the problems identified. In proposing the flow control scheme, we pursue a fundamentally different approach, which we refer to as a hop-by-hop, hybrid window and rate-based flow control scheme. As the name suggests, the rate control and window mechanism is implemented on a per-hop basis. Previous hop-by-hop protocols use either a rate control [1] or a window scheme [2] but not both in combination. While previous hop-by-hop schemes suffered from per-flow management and rate measurement complexity [1] [2], we make certain simplifications to the rate control and window design to make our design feasible for multihop wireless networks. In total, the scheme prevents congestion and reduces packet loss. While this paper is not specifically about IEEE 802.11, most of our experimental results and enhancements are based on the IEEE 802.11 DCF MAC protocol. However, the suggested framework is applicable to a general CSMA/CA MAC protocol.

II. Factors affecting throughput in multihop transmission

In this section, we diagnose and discuss some of the issues that affect throughput in a multihop wireless network when CSMA/CA MAC protocol is used. The effect of exposed node problem on throughput has been discussed in great detail in [11]. In [12], the authors have shown via simulation that throughput in CSMA/CA based multihop networks can be further degraded by the "critically exposed node problem". For the sake of completeness, we briefly reiterate this problem and further reveal three more reasons, which significantly cause the overall throughput in multihop networks to degrade.

A. The critically exposed node problem

In Fig 1, when node 7 is communicating to node 8, node 4 can initiate a transmission by sending a request to send (RTS) frame because it senses the channel as idle. However, a receiver such as node 5, will not reply since it senses the channel as busy. The location dependent interference causes disparity in the carrier sensing state between the sender (node 4) and the receiver (node 5). Due to this disparity and the use of binary exponential backoff (BEB) mechanism, node 4 will double its contention window and retry with another RTS frame. If node 4 continuously does not receive a response and the maximum retry limit is reached, the data frame will be dropped at node 4. Node 5 is termed as the "*critically exposed node*", when node 7 is transmitting. Node 4 is essentially blocked from forwarding the packet.

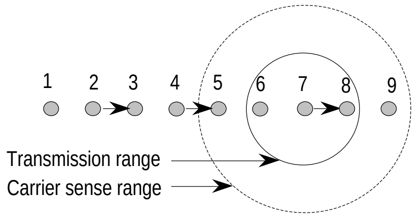

Figure 1. Chain setup with flow traveling from node 1 to node 9

B. Location dependency and uneven carrier sensing resulting in congestion

In a multihop wireless network, the location of nodes and the direction of transmission affect the throughput of a particular flow. We use figures 1 and 2 to illustrate the effect of uneven carrier sensing.

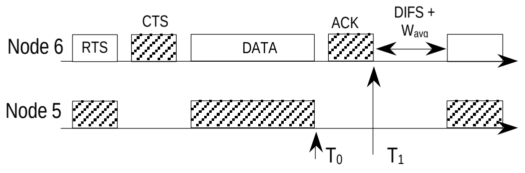

Figure 2. Uneven carrier sensing due to node location

In Fig 1, data packets are transmitted from node 1 to node 9 via the intermediate nodes. Fig 1 also shows the transmission and carrier sense ranges of node 7, when node 7 is transmitting. The intended destination of node 7 is node 8. The timelines in Fig 2 describe the reception of the signals at nodes 6 and 5, respectively, when node 7 is engaged in a 4-way, RTS-CTS-DATA-ACK handshake with node 8. The shaded frames in Fig 2 represent the carrier sense range frames received at nodes 5 and 6. When a node is within the carrier sense region but outside the transmission range of a particular sender, the received frame will be treated as noise. Therefore the Network Allocation Vector (NAV) cannot be received correctly and the node can only depend on the physical carrier sense instead of the virtual carrier sense (NAV) to perform backoff. In the setup shown in Fig 1, node 5 is outside the carrier sense region of node 8. Therefore node 5 experiences longer idle time when compared to node 6, especially during the CTS and ACK exchanges from node 8. Due to this uneven idle sensing and unequal termination times (denoted as T_0 and T_1), node 5 will start counting down its backoff counter faster than node 6. This causes node 5 to always capture the channel faster than node 6. If the offered load from node 5 is high, and this phenomenon persists, then node 6 will spend most of its time being engaged in receiving packets rather than sending them. When this occurs, the queue at node 6 will soon be exhausted and congestion will prevail. The same phenomenon will occur anywhere along the chain if the offered load is high and the idle sensing period of the upstream node is greater than the downstream node. In later sections, it will be apparent that, one of the techniques that we propose for improving throughput is designed on the basis of balancing the time spent by a node for receiving and transmitting packets.

C. False NAV

The use of virtual carrier sense or Network Allocation Vectors (NAV) was made popular in the IEEE 802.11 [13] protocol and continues to be adopted in newer protocols such as IEEE 802.15.4 [9]. Both, physical and virtual carrier-sense functions are used to determine the state of the medium. In [13], the standard states that if either function indicates a busy medium, the medium shall be considered busy; otherwise it shall be considered idle. The virtual carrier sense mechanism is achieved by distributing reservation information announcing the impending use of the medium. The exchange of RTS and CTS frames prior to the actual DATA frame is one means of distributing this medium reservation information. The RTS and CTS frames contain a "Duration" field that defines the period of time that the medium is to be reserved to transmit the actual DATA frame and the returning ACK frame. All nodes within the reception range of either sender's RTS or receiver's CTS shall obey the medium reservation.

The wrong use of NAVs in CSMA/CA MAC protocols can be detrimental to the throughput of multihop wireless networks. We use the chain setup as shown in Fig 1 to illustrate the problem that currently exists. Consider the situation when node 7 is in the midst of transmitting a DATA frame to node 8, and node 4 tries to send a RTS frame to node 5. Node 5 does not reply with a CTS frame because it is critically exposed by node 7. However, the RTS sent out by node 4 might be received by node 3. In 802.11, when a sender sends a RTS packet, it sets its NAV and informs the surrounding nodes to backoff for the entire duration of the 4-way handshake message sequence. As such, the channel will still remain reserved by the sender who had initially sent the RTS packet. If the data packet is large, the backoff period imposed by NAV can be larger than the average backoff interval chosen by the binary backoff algorithm at any backoff stage. In Fig 1, when node 2 tries to communicate to node 3, even when both nodes senses an idle channel, node 3 will not reply due to the initial NAV value sent out by node 4. Repeated RTS retries from node 2 will result in dropped packets at node 2 when the maximum retry limit is reached. Effectively, this "false NAV" indication from node 4 results in a larger exclusion area and prevents certain nodes from using the channel effectively. In some cases, this scenario could have a domino effect, thereby, impacting drastically on the throughput of multihop wireless networks.

D. Frozen MAC State

Under perfect collision avoidance, a returned CTS frame should be free from collision since the RTS sender would have reserved the channel around it. However, due to hidden terminal problem, there is a possibility for frames such as

ACKs to collide with the reception of the CTS frame. We use Fig 1 to illustrate this problem. Assuming that in the first instance, only nodes 4 and 5 are engaged in a 4-way handshake and node 4 is in the midst of sending a DATA frame to node 5. Midway through this transmission, node 7 transmits a RTS frame to node 8. Meanwhile, when node 7 is transmitting its RTS frame, node 5 completes the reception of the DATA frame and sends out an ACK frame regardless of the channel sensing state. During this period, node 8 receives the RTS frame properly and sends out a CTS frame because it senses the channel as idle. However, due to node 7's proximity to node 5, it receives the ACK frame from node 5 and the CTS frame collides at node 7. Node 7 will cease to send the DATA packet but the MAC state in node 8 anticipates the reception of a DATA frame. However, it will remain frozen in this state and will not respond to other incoming RTS frames, including those from the original sender, until the initial DATA frame interval has timed out. Such a scenario also causes blocking and results in unnecessary RTS retries.

III. Severity of the Congestion, False NAV and Frozen MAC state Problem in Multihop networks

In this section, we quantify the impact of the causes for performance limitations outlined in the previous section.

A. Metrics

Throughout this paper, we use a couple of measures to evaluate the performance of the original and newly proposed scheme: *end-to-end throughput*, (S), and *transmission cost*, (y). The *transmission cost*, (y), measures the amount of bits expended by the nodes in the system to transmit a single data bit from the source to the destination. The proposed metric is motivated by the metric proposed in [14]. However this definition is modified to cater for a multihop scenario and the overheads incurred by all the 4-way handshake frames. Therefore, the *transmission cost*, y, which gives us some indication of the energy wasted by the system is defined as the ratio of all transmitted frames in the system over the data frames received at the receiver.

B. Experiments

Using a series of experiments carried out by using NS-2 simulator, we demonstrate the problem of congestion, false NAV and frozen MAC state in multihop transmissions. All the simulations in this paper are based on the 802.11 Distributed Coordination Function (DCF) protocol with transmission rate of 1 Mbps. The transmission and carrier sense ranges are 250m and 550m, respectively. A packet capture model which only accepts packets that arrive within a short capture interval, t_c, of 4μs and ignores other received packets even with higher power signal is implemented into the original 802.11 DCF scheme. Unless stated otherwise, static routing is used in all the experiments to omit the effect of routing layer inefficiencies and accurately study the effect of MAC protocol alone on the performance of the multihop network. In addition, a fixed packet length of 1500 bytes is used in all experiments.

The first experiment was carried out by using a chain setup as shown in Fig 1. However, we reduce the number of nodes in the chain to 6 for this experiment. The data packet is sent from node 1 to node 6. The end-to-end throughput is recorded in Fig 3. The packet inter-arrival time at the source node, which is labeled as node 1 in Fig 1, is varied from 1 to 0.0001 seconds. The throughput reaches a peak of 213.1 kbps when the packet inter-arrival time at node 1 is 0.057 seconds.

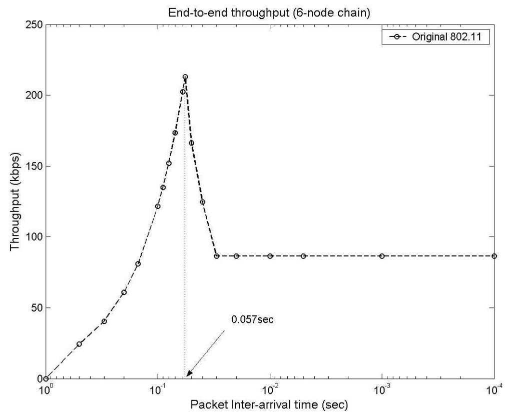

Figure 3. End-to-end throughput for chain setup

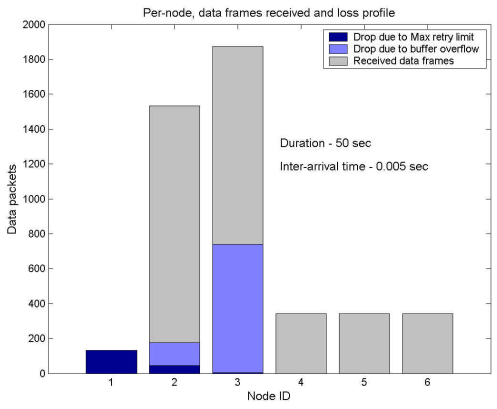

Figure 4. Number of received and dropped DATA frame for chain setup (packet inter-arrival time = 0.005 seconds)

In Fig 3, when the optimal offered load is used, the highest throughput at packet inter-arrival time of 0.057 seconds confirms the maximum attainable throughput in a chain setup as reported by [11]. However, when the packet arrival is faster than this value, the end-to-end throughput saturates at 86.94kbps. In this scenario, when the packet inter-arrival time drops below 0.057 seconds, congestion will persist. Figures 4 and 5 show the number of data frames received and dropped at various nodes along the chain for two packet inter-arrival times; 0.005 seconds and 0.057 seconds, respectively. In the first case (Fig 4), there are many dropped data packets at nodes 3 and 2 due to buffer overflow. We also notice that data packets are being dropped at node 1 and node 2 due to MAC layer maximum transmission retry limit. This phenomenon is attributed to the critically exposed interference and false NAV indication as discussed in Sections II-A and II-C above. The congestion phenomenon at nodes 2 and 3, which causes dropped packets form the buffer when the offered load exceeds a certain threshold confirms our observation, which is discussed earlier in Section II-B.

In Fig 5, there are no dropped DATA frames since the packets are scheduled by using an optimal packet inter-arrival time (0.057 seconds). In such a scenario, the deviation of the end-to-end throughput, which is 213.1 kbps, from the maximum link throughput (1 Mbps) is attributed to the exposed node effect and other overheads.

Fig 6 shows the transmission cost of the plain 802.11 MAC protocol when the chain setup is used. When the packet arrival is slower than the critical value of 0.057 seconds, the transmission cost is optimal because only a single RTS frame is consumed for every DATA frame. Under such cases, the transmission cost is recorded at 5.57. When the packet arrival is faster than the critical value, the transmission cost starts to increase due to a variety of reason discussed in Section II above. Under the heavy offered load cases, the difference in transmission cost can be as high as 210% from the optimal case.

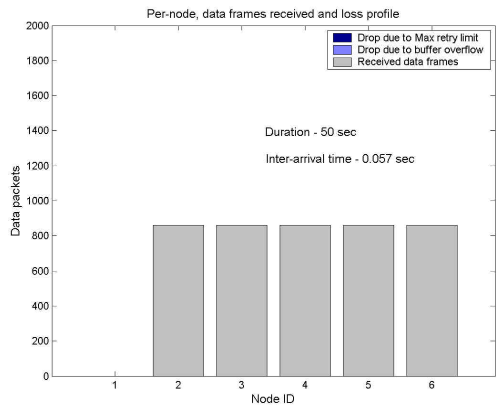

Figure 5. Number of received and dropped DATA frame for chain setup (packet inter-arrival time = 0.057 seconds)

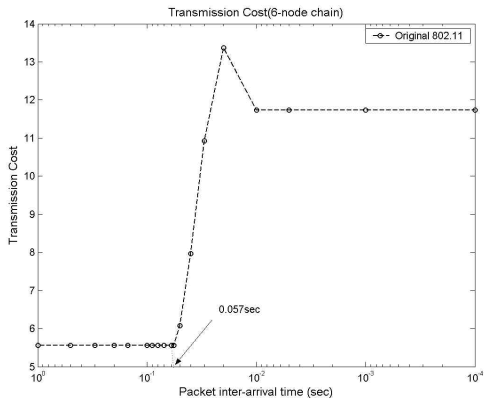

Figure 6. Transmission cost for chain setup

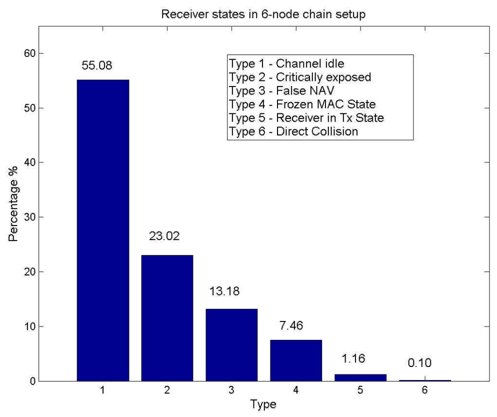

Figure 7. Percentage of various receiver states in 6-node chain setup when packet inter-arrival at node 1 is 0.005 sec.

Fig 7 shows the overall receiver responses when a RTS frame is transmitted by one of the nodes in the 6-node chain setup. The packet inter-arrival time at node 1 is set to 0.005 seconds to simulate the saturated case. The graph show that, for 55.08% of the RTS frames sent to a receiver, the receiver sees an idle channel and responds with a CTS frame. A significantly high percentage (23.02%) of the RTS frames sent to the receivers are critically exposed and the receivers will not respond with CTS frames. False NAV and frozen MAC state amounts to about 13.18% and 7.46% of the failed CTS responses, respectively. The remaining problems, which are insignificant compared to the other problems, are due to direct collision when the receiver transmits at the same instance as the sender (1.16%) and two transmission range signals colliding at the receiver (0.10%).

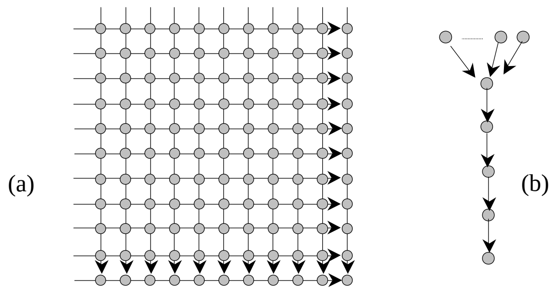

Figure 8. Grid setup and high fan-in chain setup

The above experiment only reveals the throughput and energy performance under a single end-to-end flow scenario with low node density. To demonstrate the throughput performance under multiple flow scenarios, we simulated the original 802.11 MAC in a grid topology depicted in Fig 8a where node density and channel contention is higher. The size of the grid is fixed at 1000m by 1000m, but we vary the number of nodes and distance between the rows and columns to increase the density of nodes. The flows are transmitted from the nodes in the upper row to the lowest row and nodes from the first column to the last column. In all three grid sizes shown in Table I, the number of hops to reach the destination is 5 in all end-to-end flows. In addition to the grid setup, we simulated a high fan-in scenario as shown in Fig 8b, where many single-hop links fan-in into a chain of nodes. The later scenario depicts a possible scenario in sensor networks. In both setups, we set the packet inter-arrival time of the source nodes to 0.005 seconds to simulate the condition where nodes transmit as fast as possible.

TABLE I. PERFORMANCE RESULTS FOR GRID SETUP

Grid Size	Distance between rows or cols	Transmission cost (γ)	Aggregate Throughput (kbps)
6 by 6	200 m	11.35	253.536
11 by 11	100 m	13.11	219.184
21 by 21	50 m	15.56	182.4

TABLE II. PERFORMANCE RESULTS FOR HIGH FAN-IN CHAIN SETUP

No of Sources	Transmission cost (γ)	Aggregate Throughput (kbps)
1	11.48	86.94
10	31.21	31.31
20	31.75	30.70
40	30.52	31.92

The results obtained in Table I are as expected since throughput and energy efficiency degrades as the node density and contention increases. The transmission cost is still high considering that each flow needs to travel only 5 hops to reach the destination. Table II records the performance evaluation of the topology shown in Fig 8b. The end-to-end throughput drops drastically to around 31 kbps when the number of flows going into the chain of nodes increases. The transmission cost also degrades by 548% compared to the ideal case ($\psi = 5.57$) when the number of flows is increased from 1 to 10. The experiments above demonstrate that under heavy traffic conditions and high fan-in situations, the throughput of wireless multihop networks based on CSMA/CA MAC protocol can easily degrade.

In most wireless networks, the MAC protocols and flow control schemes operate independently of each other. There have been very few research efforts on the study of an effective cross layer flow control design that increases the throughput of wireless multihop networks [28]. The MAC layer is normally isolated from the upper layers and an event such as buffer overflow is transparent to the MAC layer. Through the experiments above, we have examined some inherent MAC protocol flaws that cause unnecessary packet drops. In the later sections, we propose enhancements to the MAC protocol and introduce a novel hop-by-hop flow control scheme to improve the performance of the CSMA/CA based multihop wireless network.

IV. A FLOW CONTROL FRAMEWORK

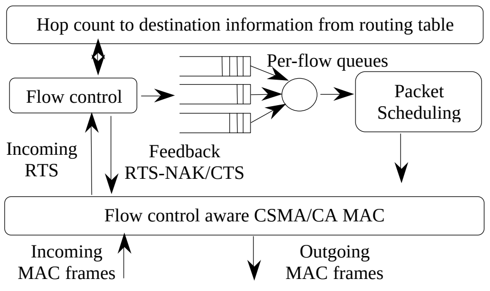

Figure 9. Flow control framework

The main objective of this flow control framework is to maximize throughput and energy efficiency of the CSMA/CA based multihop wireless network. The primary goal of the techniques that we introduce in this paper is tuned towards improving the throughput in multihop wireless networks. Since the framework is targeted for ad hoc and sensor networks, we adopt the techniques and algorithms that are efficient in terms of computations and transmissions, and thereby energy efficient. The overall flow control framework is shown in Fig 9 and consists of two major components; the enhanced CSMA/CA MAC and the hop-by-hop flow control. The two major components are explained in detail in the following sections.

Fairness of flows is another objective of our framework. We adopt a per-flow packet scheduler that tries to achieve reasonable fairness but with low processing complexity. We use the Distributed Round Robin (DRR) scheme to implement the packet scheduling scheme because it has a *O(1)* time packet processing complexity when compared to the other per-flow timestamp schedulers such as SCFQ, WF2Q, etc.

V. ENHANCED CSMA/CA MAC PROTOCOL

The CSMA/CA MAC protocol block as shown in Fig 9, is modified such that the MAC layer is aware of flow control and buffer states. Unlike the traditional stack, our MAC layer is designed to act as an admission control point to prevent eventual packet drop due to buffer overflow. In this section, we discuss the MAC enhancements that are introduced to improve the throughput and rectify the MAC inefficiencies highlighted in Section II. Specifically, we discuss how the false NAV problem should be handled and discuss the additional modification to the CSMA/CA 4-way handshake to allow interoperation of the MAC protocol with the flow control scheme.

A. *Solving the false NAV problem*

To solve the false NAV problem, we recommend that nodes apply a selective reset on the received NAV. When a node receives a NAV from certain nodes, it is expected to physically sense some activity in the channel because it is likely to be within the transmission range of either one or both of the communicating sender-receiver nodes. To solve the false NAV problem, we propose to modify the DCF protocol such that nodes that are blocked with a NAV should still try to receive transmission range packet using the normal packet capture procedure [13]. If a valid RTS frame is received correctly, the node should cancel its NAV and proceed with the normal 4-way handshake. This operation is valid since the node would not have received a valid RTS frame if there was a valid 4-way handshake ongoing in the neighborhood of the node that received the NAV. We do not however recommend that all nodes cancel their NAV by listening to the channel because resetting NAV in all nodes can result in increased interference.

B. *Modifications to the 4-way handshake*

To ensure interoperation between the MAC layer and the flow control module, we propose that the standard 4-way handshake in CSMA/CA is modified to include a RTS-NAK frame. In addition, the RTS frame is modified to carry a flow ID field, which is explained in Section VI-A. The modified 4-way handshake operates as follows. When a sender node sends a RTS frame, the receiver has the option to send a RTS-NAK frame or a CTS frame. Upon receiving the RTS, the MAC layer consults the flow control module, to check if the packet can be admitted into the node. The flow control examines the flow ID to check if a packet with the same ID exists at the receiver. If the flow violates the buffer occupancy rule, the MAC layer will reply with a RTS-NAK frame to the sender node. The buffer occupancy rule is violated when a packet that belongs to the same flow exists in the receiving node or the total buffer limit has been reached. Accordingly, there are two different RTS-NAK types to represent these two events. The RTS-NAK frame is designed as small as possible to reduce the overhead incurred by this frame. 2 bits are used to represent the different RTS-NAK types. Another 14 bits are used to carry the flow rate information, which is explained in Section VI-D. Using the same IEEE 802.11 standard, the RTS-NAK message is the same size as the CTS frame. If the flow control allows the data packet to be admitted, the 4-way handshake proceeds as per-normal.

VI. HOP-BY-HOP HYBRID WINDOW AND RATE BASED FLOW CONTROL

In the previous section, we described the enhancement to the standard 4-way handshake with additional fields and a RTS-NAK frame for the purpose of compactly carrying flow control information. In this section we discuss the hop-by-hop flow control scheme. We refer the readers to [15] for a complete taxonomy on flow and congestion control techniques. Similar to the definition used in [15], we claim that flow control is used here as a means to solve the congestion problem. Due to scalability and ease of deployment, end-to-end flow control schemes such as TCP have dominated the wired networks. Hop-by-hop flow control is normally favored over the end-to-end control scheme since it provides faster feedback response to congestion. However, the benefits of hop-by-hop schemes normally come at the expense of additional explicit messages to notify the upstream nodes about the congestion. In a wireless multihop network, if special control messages are sent over multiple hops in the reverse direction to control the rate, it can result in self-contention, which results in a sluggish system.

Hop-by-hop flow control schemes have not been popular in wired networks because of its requirement to maintain per-flow information at each node. This is understandable since the number of flows passing through a core router in the Internet is enormous, making the management of per-flow information infeasible [16]. Unsurprisingly, end-to-end flow control schemes have dominated the Internet by far. In a multihop wireless network, the number of flows is presumably more manageable if we implement some form of admission control at the nodes to restrict the number of flows passing through a node. A case for hop-by-hop flow control for wireless multihop networks has been justified in [5]. Furthermore, depending on how a flow is classified, we can further reduce the complexity of per-flow information management. Since fairness is a secondary objective in this flow control framework, we can relax the requirement of maintaining the strict per-flow information similar to that normally described in previous schemes [1] [2]. We present a lightweight flow ID management scheme that is beneficial to a computationally and energy challenged network such as ad hoc and sensor networks.

A. Flow ID representation and management

To reap the benefits of the hop-by-hop flow control and per-flow packet scheduling schemes, the process of flow identification is compulsory. However, we need an efficient method, which incurs low communication and memory overheads to represent flows. In a typical IP system, the flow is defined in an end-to-end manner and the address information that identifies the flow is normally embedded in the data packet. Hence, the entire MAC layer 4-way handshake has to be completed before the system can identify the flow. In the standard network stack without cross-layer consideration, allowing the data packet to arrive at a node before inferring if a flow has violated the buffer occupancy limit, and finally discarding the data packet is wasteful in terms of channel resources. For this reason, we need an efficient way to notify the receiver of the impending data packet before actually sending the large data packet. The IP addresses and port numbers, which are used to define the end-to-end flow, uses a total of 12 bytes. Encoding the 12 bytes into the MAC layer frames is one option but this makes the MAC frame size too large. Hence, we need a more compact flow ID representation scheme.

There are few approaches that can be used. One could design a special field to maintain a globally unique flow ID but scalability of such a technique is difficult to achieve in a multihop wireless network. The other option is to design a flow ID scheme that is unique on a per-hop and flow direction basis. This scheme is much more attractive than the globally unique flow ID scheme because the same ID can be reused by the same node on other links. However, because the total number of flows, N, that can pass through a node in an interval can be much greater than the total buffer size, R, we need to maintain a reasonable amount of flow ID records. In addition, we require a flow ID management protocol that operates between any two nodes to assign a new ID or replace old per-hop flow IDs when a node runs out of IDs. To cater for a system that prevents frequent changes in per-hop flow IDs, we then require at least $\log_2 N$ bits to represent each flow and $O(N)$ memory records for each per-hop link.

To reduce the memory storage problem attributed to flow ID records and overcome the flow ID management problem, we propose an innovative method based on hashing to represent flow IDs. In our scheme, flow IDs are represented on a per-hop basis. The IDs are generated by using a standard hashing function [17] that takes the following triplets as inputs; the unique end-to-end source and destination addresses of the flow, and per-hop source MAC address. The per-hop MAC address is used as one of the inputs for the purpose of reducing hash code conflicts due to multi-path flows originating and terminating from the same source and destination but passing through the same intermediate node. This simple but novel method obviates the need to have an additional flow ID assignment protocol.

Conflicts in the hashed codes can occur in our system if the hash code size is small. However, we argue that the system is tolerant of conflicts since per-flow fairness is a secondary objective in our flow control framework. In our scheme, we maintain a table of the triplets mentioned above only for the packets that are present in the buffer of a node at any given instance. As such, the memory requirements for storage of per-flow ID is then reduced to $O(R)$, irrespective of the number of flows passing through a node. The reduced memory requirement of this scheme is especially attractive for a sensor network environment.

In our experiments, we reserve a 10-bit field for the purpose of encoding the flow ID. The number of bits is a design parameter that has to take into consideration the statistical probability of hashing code conflicts at any given node in the system. The flow ID, which is compact, can then be encoded into the RTS frame for easy identification of the flows during the RTS-CTS phase. The benefit of this scheme is that, the sender of the RTS frame is able to query the receiver's buffer occupancy state fast by using the same MAC layer collision avoidance frames. The receiver can then positively or negatively acknowledge the buffer occupancy state for the impending flow with a CTS or a RTS-NAK frame, respectively.

B. Bandwidth-Delay Product and flow control design

We propose a new hybrid hop-by-hop flow control scheme, which utilizes a combination of static window and rate control to regulate the flow of packets. The rate control is used on a per-hop basis rather than just at the originating node. The rationale of such a scheme will be further explained in this subsection.

Before we discuss the scheme and the parameter selection in detail, we study the *bandwidth-delay product*, BDP [15], of the typical wireless multihop network. The *bandwidth-delay product* or BDP is an important parameter in flow control theory as it determines the optimal amount of packets that can be transmitted between endpoints in an uninterrupted manner. Most window based flow control scheme such as TCP's AIMD [18] and traditional sliding window protocol in effect use the BDP to calculate the optimal window size, w, for transmission between endpoints. Assuming the bottleneck service rate of a node in the path is m packets/seconds, and round trip time is measured as RTT seconds, the *bandwidth delay product* is simply given as:

$$\text{BDP} = RTT \times m \tag{1}$$

If the flow control scheme is not controlled properly and w exceeds the BDP, then the number of packets buffered at a bottleneck node becomes (w - $RTT \times m$). To reduce congestion and buffer buildup, ideally, we want to adjust w such that $w = RTT \times m$..

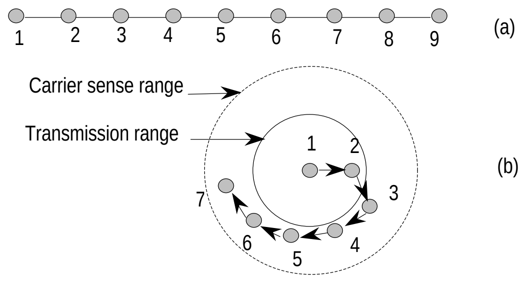

Figure 10. (a) Wired 8-hop link. (b) Spiral wireless multihop links

BDP in a wired network and wireless network has many fundamental differences. In a wired network, each intermediate node consists of more that one interface. Whereas, in a wireless multihop network, each node usually uses a single half-duplex transceiver. While the wired nodes can transmit in parallel using adjacent links, wireless nodes have to contend spatially to transmit a single frame. Fig 10a illustrates a typical chain setup using wired networks whereas Fig 1 represents the chain setup for a wireless network. In Fig 10a, packet transmission can occur simultaneously in each wired link when packets are forwarded from node 1 to node 9. Assuming negligible acknowledgment packet transmission time and a fixed data packet size, the optimal window size can be worked out as 8 packets due to the 8 hops present in the topology. Whereas, in the wireless multihop network as shown in Fig 1, parallel transmissions is a function of the transmission and carrier sensing ranges of the MAC protocol. Denoting transmission range as d_{tx} and carrier sense range as d_{cs}, by geometry we can easily show that the maximum effective throughput, S_{chain_max}, of a straight chain is upper bounded by the following expression:

$$S_{chain\_\max} = \frac{a \cdot S_{link\_\max}}{\left\lceil \frac{d_{cs}}{d_{tx}} \right\rceil + 1} \qquad (2)$$

where, α is simply a factor to account for the fixed overheads consumed by RTS, CTS and ACK frames, and other headers. S_{link_max} describes the maximum link capacity and the ceiling function, $\lceil\ \rceil$ rounds up the ratio between the carrier sense and transmission range. In a 1 Mbps IEEE 802.11b system, S_{link_max} = 1 Mbps and the ratio of d_{cs}/d_{tx} is typically 2.2 [19]. In a typical 11 Mbps system, the ratio of d_{cs}/d_{tx} is about 4.4 [19]. Using the typical 1 Mbps IEEE 802.11 ranges, it is shown in [11] that the optimal end-to-end throughput of the chain is governed by a factor of ¼. We simply refer to the factor, $\lceil d_{cs}/d_{tx} \rceil +1$ from equation (2) as the link reuse factor Q. Assuming a fixed packet size and ideal transmission scheduling, then it is easy to see that every link in a chain setup should be scheduled every Q slots, where each slot interval, T_{slot} is defined by the minimum 4-way handshake interval given as:

$$T_{slot} = T_{RTS} + T_{CTS} + T_{DATA} + T_{ACK} + 3sifs + difs + ave\_cw \qquad (3)$$

where, the *ave_cw* parameter describes the average binary exponential backoff window slots used in the first stage of the binary exponential backoff process and T_{RTS}, T_{CTS}, etc. describes the time taken to transmit the RTS, CTS, etc. frames, respectively. In other words, in a chain setup using a 1 Mbps system with d_{cs}/d_{tx} =2.2, parallel transmissions can only be carried out at every 4 hops away assuming that each transmission is carried out at maximum transmission distance and the links are in a straight line as shown in Fig 1. However, if the links are much shorter and have arbitrary direction as shown in Fig 10b, then the effective throughput, S_{chain}, of the end-to-end chain will drop below that defined in (2).

Unlike the wired case, we can show that by using equations (1) and (2), the maximum window size, w, necessary for optimal flow control in a multihop transmission is always lower than the number of hops, H, when $H^{3}2$. Assuming a negligible acknowledgement packet transmission time and wireless propagation delay, the upper bound on the window size, w_{u_bound}, for a particular flow that traverses H hops, where $H^{3}Q$ can be approximated to:

$$w_{u\_bound} = \frac{H_{count\_opt}}{\left\lceil \frac{d_{cs}}{d_{tx}} \right\rceil + 1} \qquad (4)$$

where H_{count_opt} is the hop count for the end-to-end flow assuming the chain is in a straight line and distance between nodes are at maximum transmission distance. This result gives us some clue on how the per-hop static window and rate control should be implemented at each intermediate node. Intuitively, by observing equation (4), we can conclude that it is unnecessary to allocate a total buffer space for a particular flow along the chain of nodes where this flow passes through, which exceeds w_{u_bound}. These packets will anyway cause unnecessary backlog in the intermediate nodes and prevent other nodes from accessing the network. In addition, the additional packets originating from the same flow will result in self-contention during channel access.

C. Static hop-by-hop window control

The basic hop-by-hop flow control scheme that we adopt in this framework resembles a simple static window protocol of typical first generation flow control scheme [15]. We however reuse the CSMA/CA MAC protocol's collision avoidance frames (RTS, CTS) , RTS-NAK and ACK frame as a means to implement the messaging protocol for the static window. In a multihop wireless network, it is difficult to achieve the maximum chain throughput, S_{chain_max}, since there will be other contending nodes that effectively reduce the throughput of the end-to-end link. In addition, a packet that experiences frequent blocking will have a longer average transmission period than that given in equation (3) to send a packet successfully, due to multiple RTS retries and long binary exponential backoff periods.

Tuning the nodes to use the exact window size requires constant measurement and feedback of the round trip time or flow rate, which is too resource consuming for an ad hoc or sensor network. For this reason and to keep the design simple, we simply use the upper bound of the window size given by equation (4) to design the flow control scheme.

To simplify analysis and design of the flow control scheme, we assume that nodes transmit equal length packets and the queue length at each node is measured in terms of packets. The design and parameters can be easily converted for a variable packet length system and a queue length measured in bytes. Averaging w_{u_bound} over the intermediate nodes, we note that each node should only hold a fraction of the packet. However, since the queue size is measured in packets, it is impractical to store a fraction of the packet. Therefore, we limit the static window size for each flow at each intermediate node to a single packet. We then rely on the rate control part of this hybrid hop-by-hop flow control scheme to spatially spread the packets. Since the per-hop queue used to serve a flow is limited to a fixed window size of one unit, we try to spread the packet distributed on the intermediate nodes such that they are placed at every Q^{th} link. By controlling the rate control part properly, we can effectively limit the equivalent flow control window size of a flow to w_{u_bound} along the links. The basic hop-by-hop flow control scheme with the static single-unit window alone will simply be referred to as the "Static H-b-H window" scheme.

D. Rate control

To realize the spreading of the packets or the rate control scheme, we implement a delay mechanism at each node. This rate control scheme cannot operate independently and needs to be supplemented with the basic static window scheme discussed in Section VI-C above. This is necessary because the rate control scheme requires some form of binary feedback on congestion from the hop-by-hop static window scheme. The operation of the rate control scheme is captured in the algorithm described in figures 11 to 13, which is expressed using a C-style pseudo code. For convenience, the notation and terminology used to describe the algorithm are summarized in Table III below.

TABLE III. NOTATIONS

d	Destination node for flow f
s	Source node for flow f
Pkt_f^m	The m^{th} packet for flow f
$hops(i,j)$	Number of hops from node i to node j
Timer(T)	Timer function with count down period of T seconds
Q	System wide link reuse factor (as explained in Section VI-B)
T_{slot}	Average 4-way handshake period to transmit a data packet as given by equation (3)
f.record	Flow record for flow f
$f.T_{prev\_delay}^{m-1}$	Previous delay time used by the *(m-1)*th packet of flow f. This is the time that must elapse for the m^{th} packet of flow f before it is scheduled for transmission after the *(m-1)*th packet has exited the node.
$Pkt_f^m.delay$	Current delay assigned to the m^{th} packet of flow f at a certain node in multiples of T_{slot}.
$Pkt_f^m.base.delay$	Base delay assigned to the m^{th} packet of flow f based on the remainder hops to destination
Schedule(*Pkt*, *T*)	A function to schedule the transmission of a packet, *Pkt*, after a delay of T seconds.

Unlike previous hop-by-hop flow control schemes, our scheme is unique because it does not necessitate the storage of all per-flow information for packets that transit at each intermediate node. However, we maintain flow information for packets that are currently residing at a node and for those that have exited a node, for a limited period. After this limited period, the flow record is flushed from the intermediate node's memory. However, like most of the rate control schemes, the historical flow rate at each node is crucial for optimally controlling the flow and reducing congestion. To retain the rate information of a flow and at the same time reduce the requirement of per-flow rate information storage at each node, we encode the flow rate on a given link, l, into the data packet that is forwarded to the next hop. In this way, when an upstream node tries to admit a packet originating from the same flow on link l and a packet still exist in the downstream node's buffer, the last used flow rate information for link l can be obtained from the downstream packet. The rate value is then encoded in the RTS-NAK frame and passed back to the upstream node. Since the rate information is coded in multiples of T_{slot}, the number of bits required to represent the rate can be compactly encoded in the RTS-NAK frame. The rate information can then be used to schedule the transmission of the data packet from the upstream node to the downstream node.

```
Receive( $Pkt_f^m$ )      at node i
{
1    /* initialize basic packet delay */
2    if (f.record ≡ False){
3         if (hops(i,d) ≥ Q)
4              $Pkt_f^m.base.delay \leftarrow (Q-1)\times T_{slot};$
5         else
6              $Pkt_f^m.base.delay \leftarrow (hops(i,d)-1)\times T_{slot};$
7    }
8    if (f.record ≡ False) {
9         Encode $Pkt_f^m$ with $Pkt_f^m.base.delay$
10        $Pkt_f^m.delay \leftarrow Pkt_f^m.base.delay;$
11        Schedule( $Pkt_f^m$ ,0);
12   }
13   else if (f.record ≡ True and Timer( $f.T_{prev\_delay}^{m-1}$ ) not expired) {
14        Encode $Pkt_f^m$ with $f.T_{prev\_delay}^{m-1}$
15        $Pkt_f^m.delay \leftarrow f.T_{prev\_delay}^{m-1}$ ;
16        Schedule( $Pkt_f^m$ , remainder of Timer( $f.T_{prev\_delay}^{m-1}$ ));
17   }
} /* End of Receive */
```

Figure 11. Receive function

Fig 11 describes the receiving function of the algorithm. A node, which receives a transit packet, first checks if a flow record for flow f, f.record, exists. The f.record basically stores the time that must elapse, $f.T_{prev\_delay}^{m-1}$, in between the transmission of packets (*m-1*) and m of flow f, along a link l. The f.record only exist at a node for a limited period given by $f.T_{prev\_delay}^{m-1}$, after the (*m-1*)th packet has left the node. If a record does not exist, it then proceeds to schedule the packet for transmission immediately. Before transmitting, the delay in multiples of T_{slot} that corresponds to the rate in which flow f passes through link l is encoded into the data packet. In lines 3 to 6, the number of hops to the destination is checked to compute the base delay that should be used. The delaying technique that we design gives the receiver node and other contending nodes along the downstream path ample time to forward packets that belong to the same flow. Considering this, it is intuitive to schedule the packets for every Q^{th} slot when there is Q number of links ahead. However, when the remaining hop count from node i to destination node, d, given by *hops(i, d)* is lower than Q, or an end-to-end flow has fewer than Q hops, then we need to regulate the rate control such that these flows are not unnecessarily penalized. For this reason, we apply the following delay rule at every intermediate node by considering the remainder hop-count to destination.

$$base.delay = \begin{cases} (Q-1)\cdot T_{slot} & \text{if } hops(i,d) \geq Q \\ (hops(i,d)-1)\cdot T_{slot} & \text{if } hops(i,d) < Q \end{cases} \quad (5)$$

In (5), *base.delay* denotes the base delay to be imposed for a flow at any given node. In Fig 9, the number of hops to destination is obtained from the routing layer. In lines 13 to 16 of Fig 11, the node waits for the timer, Timer($f.T^{m-1}_{prev\_delay}$), to expire if *f*.record exists before scheduling the packet for transmission. If *f*.record exists, the current delay period, which is stored in *f*.record, is encoded into the data packet before transmission.

The Schedule function in Fig 11 invokes the Transmit function shown in Fig 12. When an upstream node receives a RTS-NAK frame from a downstream node indicating that a packet of the same flow ID is still present at the downstream node, the upstream node will extract the delay information encoded in the RTS-NAK. This is described in line 2 of Fig 12. The upstream node will then select the highest delay value between the last used delay, Pkt^m_f *.delay*, and the delay information encoded in the RTS-NAK and additively increases the delay interval by a single T_{slot}. Notice that the last used delay value can be higher than the delay value encoded in the RTS-NAK due to multiple RTS-NAK replies for the same data packet. The upstream node will then restart the delay timer for transmitting the packet again.

```
Transmit( $Pkt^m_f$ )   at node $i$
{
1    if (received a RTS-NAK) {
2            Extract $Pkt^{m-1}_f$ .delay from RTS-NAK frame
3            if ( $Pkt^m_f$ .delay > $Pkt^{m-1}_f$ .delay)
4                    $Pkt^m_f$ .delay ← $Pkt^m_f$ .delay + $T_{slot}$;
5            else
                     $Pkt^m_f$ .delay ← $Pkt^{m-1}_f$ .delay + $T_{slot}$;
6        Encode $Pkt^m_f$ with $Pkt^m_f$ .delay
7        Schedule( $Pkt^m_f$ , Timer( $Pkt^m_f$ .delay));
8    }
9    else if (received an ACK) {
10       $f.T^m_{prev\_delay}$ ← $Pkt^m_f$ .delay;
11       $f.T^m_{prev\_delay}$ = max[ $f.T^m_{prev\_delay}$ -$T_{slot}$, $Pkt^m_f$ .base.delay];
12       f.record ← Store( $f.T^m_{prev\_delay}$ );
13       Start Timer( $f.T^m_{prev\_delay}$ );
14   }
}/* End of Transmit */
```

Figure 12. Transmit function

```
Cleanup ( ) at node $i$
{
1    if(Timer( $f.T^m_{prev\_delay}$ ) ≡ 0 and $Pkt^{m+1}_f$ ≡ 0)
2        f.record = False;
} /* End of Transmit */
```

Figure 13. Cleanup function

If an upstream node receives an ACK frame as a result of successfully transmitting the *m*th data packet, the upstream node will then setup the delay timer for the (*m+1*)th data packet. Lines 9 to 14 of Fig 12 describe this procedure. The upstream node will decrement the last used delay by T_{slot} if the delay period used is greater than the base delay given by equation (5) and store it in *f*.record. Else, the upstream node will simply use the base delay given by equation (5) and store this in *f*.record. The node will then set a timer based on the selected delay for the same flow *f*. Note that immediately after receiving the ACK, the upstream node will not have a packet with the same flow ID because the window size is a single unit but any packet that is received eventually will have to adhere to this delay. If the upstream node does not receive a packet of the same flow ID within this delay period, then this flow information is removed from the memory. This operation is described in Fig 13. This operation reduces the memory storage complexity associated with storage of per-flow information. The reason for decrementing the delay by a single slot in line 11 of Fig 12 is to test if the sending rate can be increased. Line 11 also limits the rate to the achievable upper bound derived from equations (2) and (4).

In this paper, we have not suggested any special method to solve the critically exposed node problem; therefore some flows will still experience excessive blocking under normal circumstances due to unfavorable location of nodes. Some of the packets will also be dropped after experiencing the maximum retry limit. The simple additive delay increase method mentioned above will regulate the flows passing through such links but it does not penalize them. As a result of this, the flow control framework proposed in this paper still experiences dropped packet due to critically exposed interference. To reduce the lost packets due to such interference, one could possibly modify the packet scheduler described in Section IV to a weighted scheme that penalizes such flows.

In contrast to our per-hop rate control scheme, one could design a source node rate control mechanism [6] to regulate the flows in the same manner as described above but a rate control mechanism on a per-hop basis is more beneficial than a single rate control mechanism at the source. The reason for this is because a source rate control can only regulate the flow at the starting point of the flow. Contention and bottlenecks in the downstream nodes will quickly cause the packets to be clumped together at each node along the downstream path, further resulting in self-contention and reduced throughput.

VII. Simulation Results

TABLE IV. SUMMARY OF SCHEMES

Schemes	Remarks
NAV repair	The selective NAV repair scheme proposed in Section V-A
Rate Control	The hop-by-hop rate control as described in Section VI-D
Static H-b-H Window	A static per-flow window of size 1 at each node
No Congestion Control or Original 802.11 MAC	The original stack with plain 802.11 DCF and FIFO queuing

To demonstrate the effectiveness of our approach, we used NS-2 simulator to obtain simulation results. We modified the IEEE 802.11 DCF protocol to include the enhancements to the 4-way handshake and the solution to the false NAV problem. In addition, we developed additional clear channel assessment modes (CCA) to handle the proper carrier sensing states, which is absent in NS-2. All the performance results obtained in this paper are based on CCA-2 [13] mode. We also implemented our flow control scheme and the MAC layer enhancements in a modular fashion such that different modules can be combined in different combinations to form different strategies. The various modular schemes that form the various strategies are summarized in Table IV. In addition, the metrics that were described in Section III are use to evaluate the performance of the various strategies. All simulations were carried out by using a fixed data packet size of 1500 bytes.

We evaluate the various strategies by using four distinct topologies. The first is a simple chain setup as shown in Fig 1, but with a single flow originating from node 1 and terminating at node 6. This simple setup has also been shown in Section III to suffer from congestion. In the second setup, we examine a directed 5-chain links fan-in situation as shown in Fig 14, where several flows traveling along different chain of nodes for some distance converge into a single chain of nodes. In Fig 14, the nodes denoted as S are the sources whereas the node denoted as D is the destination node. Next, we show the performance of the high fan-in single-hop links that converge into a single chain as shown in Fig 8b. Finally, we evaluate the performance of our scheme under the grid setup with multiple cross flows, which is shown in Fig 8a and described in Section III-B.

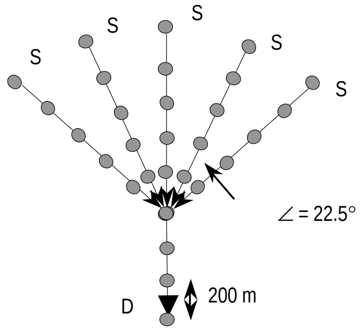

Figure 14. directed 5-chain links fan-in

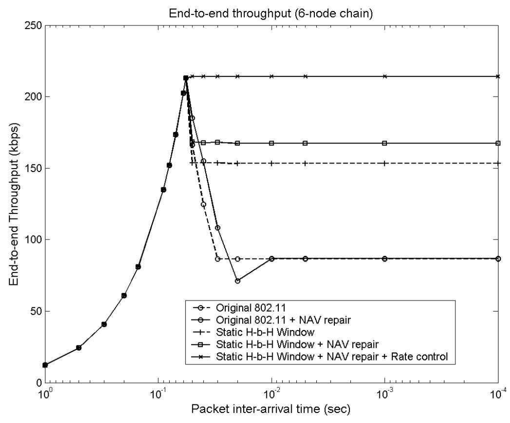

Figure 15. Throughput perfornance – chain setup

Fig 15 shows the throughput performance of the various flow control strategies when the single-flow chain setup is used. When packet arrival is slow, the behavior of all the schemes is the same. However, when packet arrival is faster, there are significant differences in the throughput performances. Fig 15 shows that the combination of the static window and rate control is sufficient to ensure that the throughput is optimal and tallies with equation (2). In this particular setup, using the rate control and static window negates the effect of false NAV; therefore the addition of NAV repair does not show any improvement. When all three schemes are used in combination, the throughput exceeds the "No Congestion Control" scheme by a factor of 2.47. When the static window is used alone, the throughput performance is increased by a factor of 1.77. Additional improvement to throughput is noticed when NAV repair is used with the hop-by-hop static window alone.

Fig 16 shows the transmission cost obtained by using the chain setup. The use of the three schemes in combination keeps the energy efficiency to the optimal value of 5.57. This corresponds to the transmission that occurs along 5 hops of the chain setup. In the "No Congestion Control" scheme, the transmission cost can exceed the optimal value by a factor of 2.4. This shows that significant amount of energy can be wasted when the flow is pumped into the chain as fast as possible and no congestion control is used. Fig 16 also shows that, significant saving of energy is possible even if the plain static window is used alone.

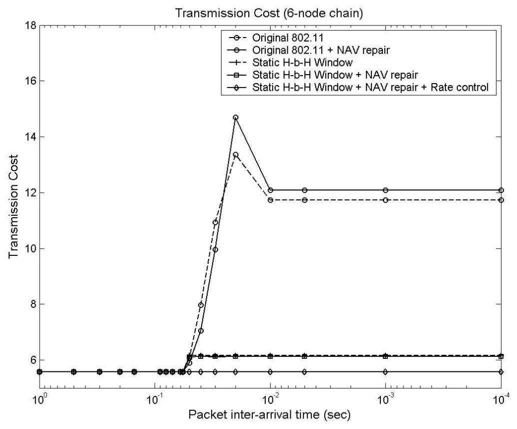

Figure 16. Transmission cost – chain setup

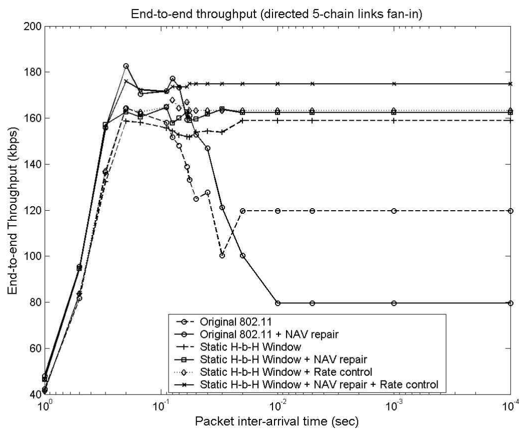

Figure 17. Throughput performance – directed 5 chain links fan-in

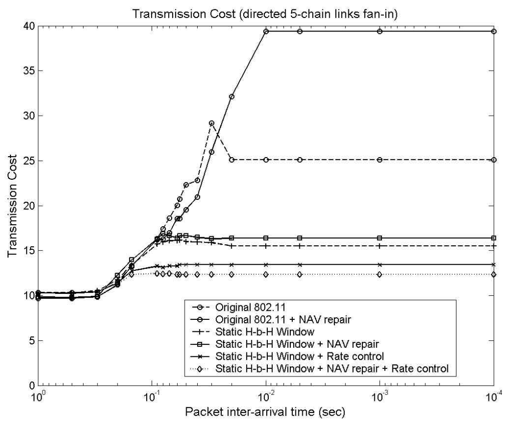

Figure 18. Transmission cost – directed 5-chain links fan-in

The throughput performance and transmission cost of the directed 5 chain links fan-in setup (as shown in Fig 14) is shown in figures 17 and 18. It is interesting to note that, in this setup, when the NAV repair is used with the plain 802.11

MAC, the throughput performance during faster packet arrival actually degrades when compared to the original 802.11 MAC. This occurs because the NAV repair employed at a node actually has the effect of improving the MAC layer service time used for receiving incoming frames. NAV repair also reduces the number of packets that are dropped at a sender due to maximum retry limit because receivers are not blocked due to false NAV. However this improvement in receiving time is achieved at the expense of reducing the service time available for transmitting frames. Therefore when NAV repair is used without flow control, more packets can be sent by the source but are later dropped due to buffer overflow at certain nodes. The simulation traces for the 5 chain links fan-in setup used in Fig 14 confirm this observation. During faster packet arrival, the total packet dropped due to buffer overflow and maximum retry limit for the original MAC with NAV repair only, exceeds the original MAC by 17.6%. However, the packet drop due to buffer overflow alone, for the original MAC with NAV repair, exceeds the original MAC by 30%. The dropped packets due to buffer overflow for this particular setup occur mainly at the 2nd and 3rd node of each end-to-end flow. As a result of reduced service time for transmitting, fewer packets are successfully transmitted to the final destination.

When the NAV repair is used in combination with rate control and static window, it actually contributes to throughput improvements. Consistent with the chain setup, throughput and energy improvement can be significantly improved by just using the static window alone. However, the combination of the NAV repair, static window and rate control yields the best performance.

TABLE V. PERFORMANCE OF FLOW CONTROL FRAMEWORK USING GRID SETUP

	No Congestion Control		Static Window + NAV repair + Rate control	
Grid Size	ψ	Throughput (kbps)	ψ	Throughput (kbps)
6 by 6	11.35	253.536	6.61	400.97
11 by 11	13.11	219.184	7.94	324.67
21 by 21	15.56	182.4	9.28	270.56

TABLE VI. PERFORMANCE OF FLOW CONTROL FRAMEWORK USING HIGH FAN-IN CHAIN SETUP

No. of Sources	No Congestion Control		Static Window + NAV repair + Rate control	
	ψ	Throughput (kbps)	ψ	Throughput (kbps)
1	11.48	86.94	5.57	214. 01
10	31.21	31.31	6.07	174.89
20	31.75	30.70	6.13	172.67
40	30.52	31.92	6.26	170.24

Table V shows the performance improvement of the complete flow control scheme when the grid setup as shown in Fig 8a is used and when sources transmit as fast as possible. The performance improvement for throughput and transmission cost is not as high when compared to the other two setups described previously but still shows significant improvement. Table VI shows the improvement when the high fan-in chain setup as shown in Fig 8b is used. The results collected in Table VI are based on sources transmitting as fast as possible. Using the complete flow control framework, tremendous improvement in throughput and energy efficiency can be obtained when the number of sources is high. When the number of sources in the high fan-in chain setup is 10, the energy efficiency and throughput improves by factors of 5.14 and 5.58, respectively, when the complete flow control framework is used.

VIII. RELATED WORKS

Flow control in wired networks have been proposed and studied for several decades with end-to-end flow/congestion control schemes dominating the earlier designs [18] [21] [22]. Some of these designs have been adopted into the infamous TCP protocol suite [18]. Due to low complexity in design, scalability and ease of deployment, end-to-end flow control schemes such as TCP have dominated the wired networks. However, as pointed out in [20] [24-28], end-to-end flow control such as TCP fails to support high utilization of bandwidth in multihop wireless networks. The authors in [27] have specifically investigated the effects of mobility and further proposed an explicit link failure notification (ELFN) to help TCP differentiate between congestion and link failure losses. The authors in [26] have also closely studied the interactions between TCP and the underlying MAC protocol such as 802.11.

Hop-by-hop flow control started to emerge in the early 90's [1] [2] [4]. Hop-by-hop flow is usually favored because it provides faster feedback response to congestion. However, the benefits of hop-by-hop schemes normally come at the expense of additional explicit notification messages [1] [6] to notify the upstream nodes about the congestion and maintenance of per-flow information at each node [1] [2] [6]. Recently, hop-by-hop flow control schemes have been suggested for use in ad hoc networks [5] [6] and sensor networks [7] [8]. Hop-by-hop schemes typically use either a rate control or a dynamic window to implement the flow control. The scheme proposed by [1] is a rate-based flow control implemented on a per-hop basis. Nodes typically measure the rate and buffer occupancy of flows passing though it and explicitly passes this information to upstream nodes. Each node is responsible for throttling the flows at the onset of congestion. In [6] [5], the schemes defer as the explicit feedback is sent to the main source to control the source rate. These schemes can suffer from scalability issues because of the expensive resources required to maintain per-flow information and compute the actual rate of the flows. The scheme proposed by [2], which is known as the credit-based flow control is a typical dynamic window scheme implemented on a per-hop basis. A downstream receiver monitors queue lengths of each flow and determines the number of packets or “credit” an upstream sender can transmit on a link. The sender transmits only as many packets as allowed by the credit. The scheme essentially monitors the outgoing links, determines the link round trip times and allocates the buffer evenly to flows passing through the node. Periodic feedback is required to indicate the available credits to the upstream nodes. Similar to [1], this scheme requires per-flow management and dedicated buffers, which is not scalable in ad hoc or sensor networks. In addition, the schemes proposed in [1] [2] [6] do not promote spatial spreading of packets belonging to the same flow, which could result in self-contention and reduced throughput at the MAC layer.

The schemes proposed in [7] [8] are examples of recent efforts in designing lightweight flow control schemes specially targeted for sensor networks. This scheme monitors the buffer occupancy level and uses a binary value to indicate congestion. The binary congestion value, which is coded in the data frame, is then broadcasted to upstream (child) node to stop the transmission of frames. The upstream (child) node, which hears this binary congestion value, will similarly broadcast this value upstream till it reaches the source. In addition to this simple hop-by-hop flow control, the scheme in [7] implements a per-hop rate control which controls the rate of upstream nodes. The sending rate is computed based on actively listening and computing the rate of the transit packets forwarded by the downstream nodes. In total, the scheme resembles a rate-based, hop-by-hop flow control scheme. The disadvantage of this technique is that if per-flow management is not considered, then the congestion bit is essentially propagated to all source nodes, which are serviced by the congested parent node. This could restrict some of the upstream nodes from forwarding packets in other directions if other path exists. The scheme in [8] on the other hand implements a hop-by-hop flow control scheme with source rate control. In [8], congestion is detected by listening to the channel activity. Similar to [7], the congestion is signaled to the upstream nodes. Similar to [7], this scheme is suitable for sensor networks where most data originate from source and mostly travel in a single direction towards a sink.

In contrast to the schemes described in [7] [8], our flow control scheme is targeted towards a more general multihop wireless network which typically have flows moving is arbitrary directions. As such, the distinction of flows is still required. The hop-by-hop flow control scheme that we describe in this paper differs significantly from other hop-by-hop schemes mentioned above. In our scheme, we introduce a hybrid per-hop static window and rate control scheme to dynamically adjust the flow rate.

IX. CONCLUSION

In this paper, we have carefully studied the CSMA/CA MAC protocol based on 802.11 DCF to identify some of the main problems that cause congestion and degradation in throughput and energy efficiency. We have categorically identified the major problems such as “False NAV” and “Frozen MAC State” and proposed a solution to the “False NAV” problem. This paper also presents a unique hybrid hop-by-hop flow control scheme, which uses per-hop static window and rate control to regulate congestion in multihop wireless networks. We proposed a flow control framework by coupling the hop-by-hop hybrid flow control and the false NAV enhancement. The hop-by-hop flow control scheme leverages on the CSMA/CA based MAC protocol by utilizing the MAC frames used for collision avoidance. The flow control scheme itself is a zero loss scheme since buffer occupancy is always monitored and fed back to the upstream nodes using the MAC layer’s collision avoidance frames. Unlike the traditional hop-by-hop flow control scheme proposed previously which suffers from per-flow maintenance of all flows passing through a node; our scheme is designed to reduce per-flow maintenance. We introduce a novel technique for storage and sharing of flow rate information. We encode the rate information of a link used by a particular packet into the packet and pass it to the downstream node. This rate information can then be extracted for eventual use by the upstream node. In addition to this, we introduce a novel technique for flow ID representation based on hashing. This simple but novel method eliminates the need for a complex flow ID assignment protocol and a large flow ID field to represent flows. The flow control scheme proposed in this paper has several benefits when compared to an end-to-end window scheme such as TCP or a pure rate control hop-by-hop scheme. While the number of packets pumped into the system by a source can be the same in all these schemes, we use a per-hop rate control scheme in combination with a static single packet window that tries to spatially distribute the packets along the links to avoid self contention among flows originating from the same source. In TCP, clumping of packets belonging to the same flow can occur, resulting in congestion and buffer overflow. Our flow control framework demonstrates that throughput and energy efficiency can be improved tremendously in simple topologies and topologies that demonstrate high congestion.

REFERENCES

[1] P. P. Mishra and H. Kanakia, “A hop by hop rate based congestion control scheme,” in *ACM SIGCOMM*, August 1992.

[2] H. T. Kung, T. Blackwell, and A. Chapman, “Credit-based flow control for ATM networks: Credit update protocol, adaptive credit allocation and statistical multiplexing,” in *ACM SIGCOMM*, 1994.

[3] D. Katabi, M. Handley, and C. Rohrs, “Internet Congestion Control for Future High Bandwidth-Delay Product Environments,” in *ACM SIGCOMM*, 2002.

[4] C. Ozveren, R. Simcoe, G. Varghese: "Reliable and Efficient Hop-by-Hop Flow Control ", *IEEE JSAC*, vol. 13, no. 4, May 1995, pp. 642-650.

[5] Y. Yi, S. Shakkottai, “Hop-by-hop Congestion Control over a Wireless Multi-hop Network”, in *IEEE INFOCOM*, 2004.

[6] K. Chen, K. Nahrstedt, N. Vaidya, “The Utility of Explicit Rate-Based Flow Control in Mobile Ad Hoc Networks,” in *IEEE WCNC*, 2004.

[7] B. Hull, K. Jamieson, H. Balakrishnan,”Mitigating Congestion in Wireless Sensor Networks", in *ACM SenSys*, 2004,

[8] C. Y. Wan, S. Eisenman, A. Campbell, “CODA: Congestion Detection and Avoidance in Sensor Networks”, in *ACM Sensys*, 2003.

[9] http://www.ieee802.org/15/

[10] W. Ye, J. Heidemann, and D. Estrin, “An energy-efficient MAC protocol for wireless sensor networks,” in *IEEE INFOCOM*, 2002.

[11] J. Li, C. Blake, D. S. De Couto, H. I. Lee, and R. Morris, “Capacity of ad hoc wireless networks,” in *ACM MOBICOM*, 2001.

[12] J. Pathmasuntharam, A. Das, A. K. Gupta, “Channel Assignment for Nullifying the Critically Exposed Node Problem in Ad Hoc Wireless Network”, in *IEEE SECON*, 2004.

[13] IEEE Std 802.11b. Wireless LAN Medium Access Control (MAC) and Physical Layer *(PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band*, IEEE, 1999.

[14] J. Zhao, R. Govindan, “Understanding Packet Delivery Performance in Dense Wireless Sensor Networks,” in *ACM Sensys*, 2003.

[15] S. Keshav, An Engineering Approach to Computer Networking. Reading, MA: Addison Wesley, 1998.

[16] R. Jain, “Congestion control and traffic management in ATM networks: recent advances and a survey,” *ATM Forum Contribution 95-0177*, 1995.

[17] R. Jain, “A Comparison of Hashing Schemes for Address Lookup in Computer Networks” , Tech. Rpt.-593, DEC, February 1989.

[18] V. Jacobson, “Congestion Avoidance and Control”, in *ACM SIGCOMM*, 1988.

[19] B. Awerbuch, D. Holmer, H. Rubens, “High Throughput Route Selection in Multi-rate Ad Hoc Wireless Networks”, *IFIP WONS*, 2004.

[20] S. Xu, T. Saadawi “Does the IEEE 802.11 MAC Protocol work well in multihop wireless ad hoc networks?,” *in IEEE Comms Magazine*, 2001.

[21] D. Mitra, J.B. Seery, “Dynamic adaptive windows for high speed data networks: Theory and simulations”, in *ACM SIGCOMM*, 1990.

[22] K. K. Ramakrishnan and R. Jain, “A Binary Feedback Scheme for Congestion Avoidance in Computer Networks”, in *ACM Trans. on Comp. Sys. Vol. 8*, no. 2, pp 158-181, May 1990.

[23] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, “System architecture directions for network sensors,” in ASPLOS-IX, 2000.

[24] H. Balakrishnan, V. Padmanabhan, and R. H. Katz, "The effects of asymmetry on TCP performance," in *ACM/IEEE MOBICOM*, 1997.

[25] J. Liu and S. Singh, “ATCP: TCP for mobile ad hoc networks”, in *Proc. of IEEE JSAC*, 19(7):1300–1315, 2001.

[26] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, “The Impact of Multihop Wireless Channel on TCP Throughput and Loss”, in *IEEE INFOCOM*, 2003.

[27] G. Holland and N. Vaidya ,“Analysis of TCP performance over mobile ad hoc networks.” in *ACM MOBICOM*, 1999.

[28] X. Yu, "Improving TCP Performance over Mobile Ad Hoc Networks by Exploiting Cross-Layer Information Awareness," in *ACM MOBICOM*, 2004.